Jürg Gutknecht, SI und ETH Zürich, April 2014

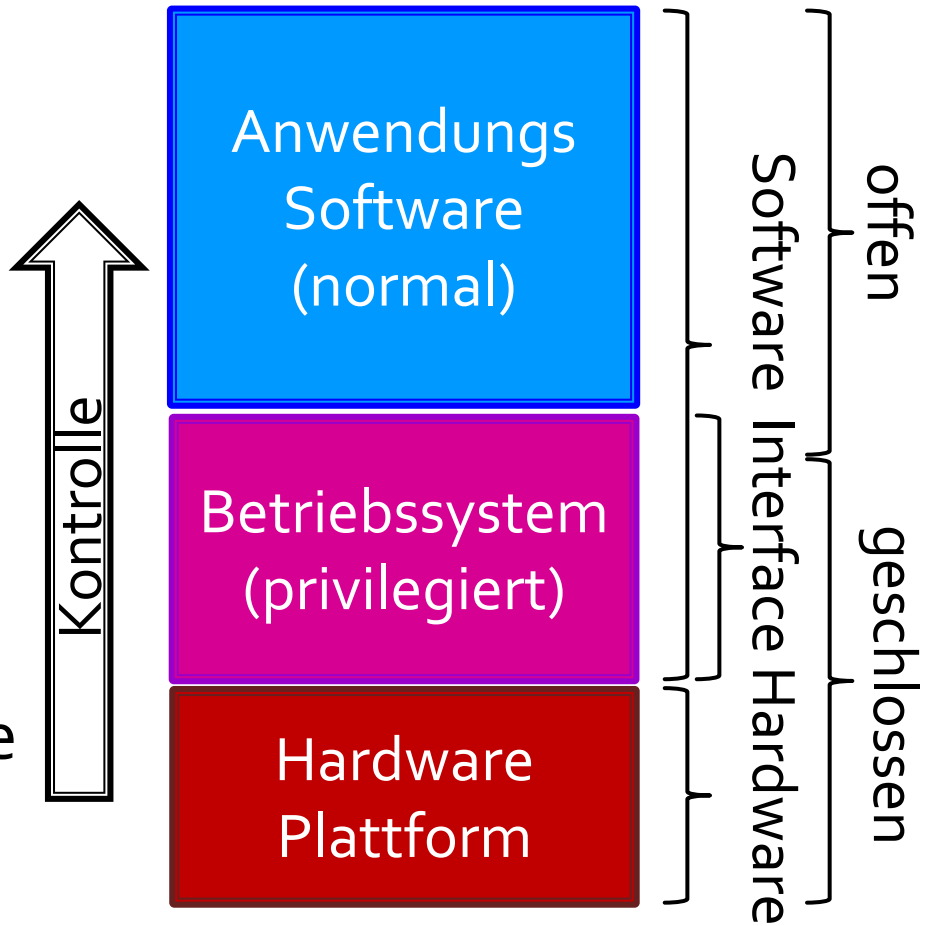
Cyber Überwachung Panel

Arten von Attacken

Art der Attacke	Ziel	Methode	Hilfsmittel	Abwehr
passiv	Datendiebstahl	Auslesen des Speichers des Opfercomputers	«Snooping»	TPM, Secure (Co-) Prozessor
		Abhören Datenverkehr • im Opfercomputer • im Computernetzwerk		
aktiv	Vortäuschung	Modifikation des Datenverkehrs	Modchip	
	Ausführung gewünschter Aktivitäten	Ausführung von Daten (z. B. ActiveX, Applets, Javascript)	Webapplikation	Interpretation, Sandboxing, Virtualisierung
		Umleitung des Steuerflusses	Stack hijacking	Execute ⊕ Write etc.

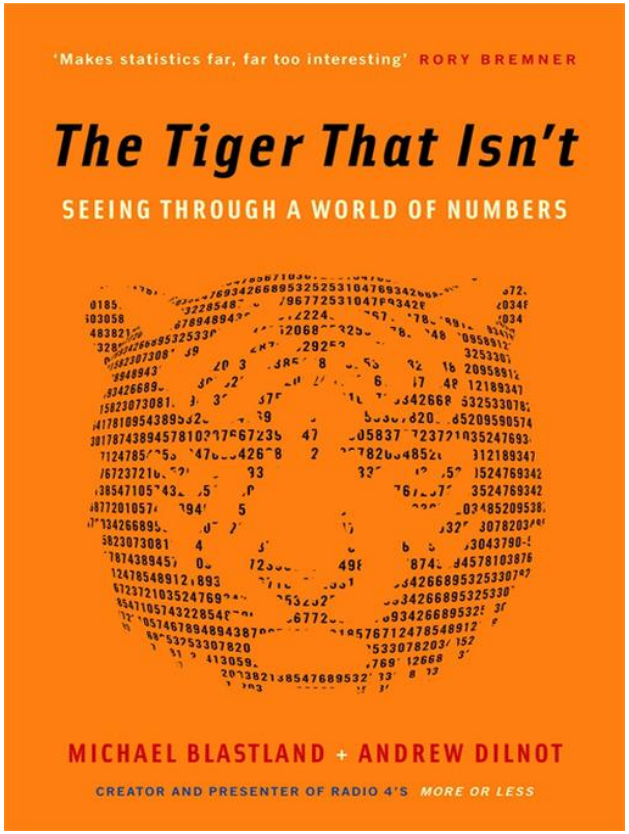
Grundstruktur von Computern

- Computer sind in drei Schichten strukturierte Universalmaschinen
- Jede untere Schicht hat vollständige Kontrolle über die jeweils oberen Schichten
- Die unteren beiden Schichten sind geschlossen, die oberste ist offen

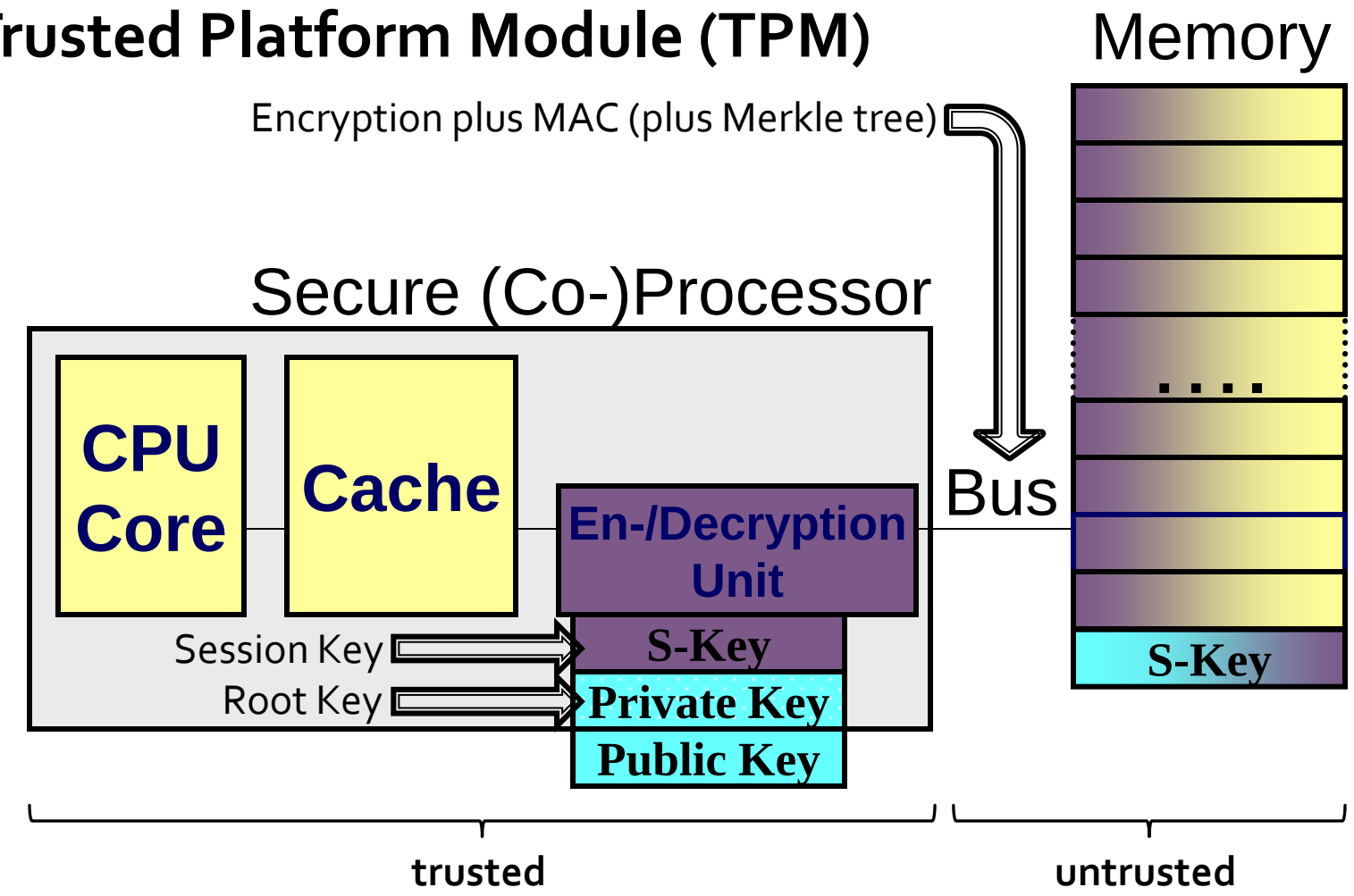


Höheres Gefahrenpotenzial

- Zufallskorrelationen in Big Data

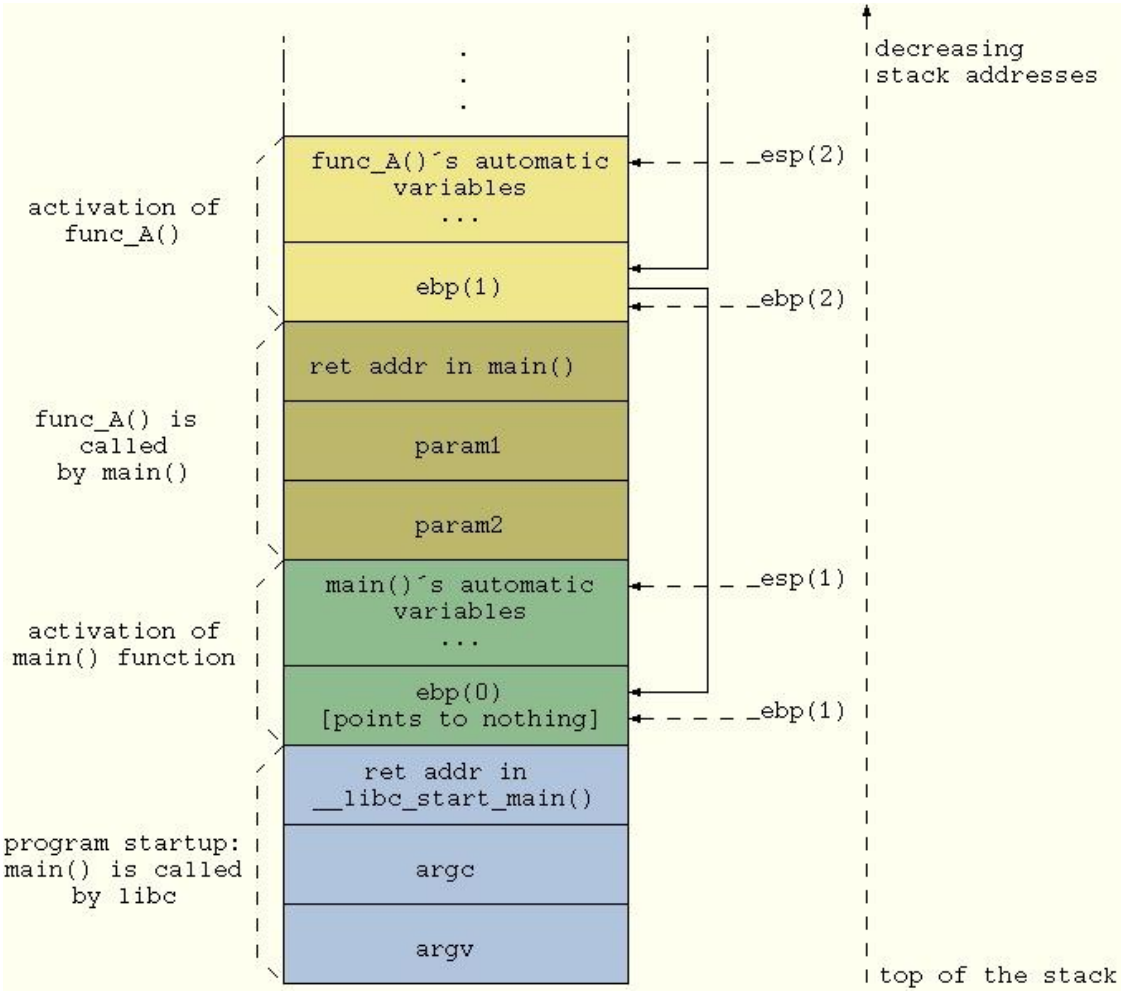


Hardware Ebene



Software Ebene

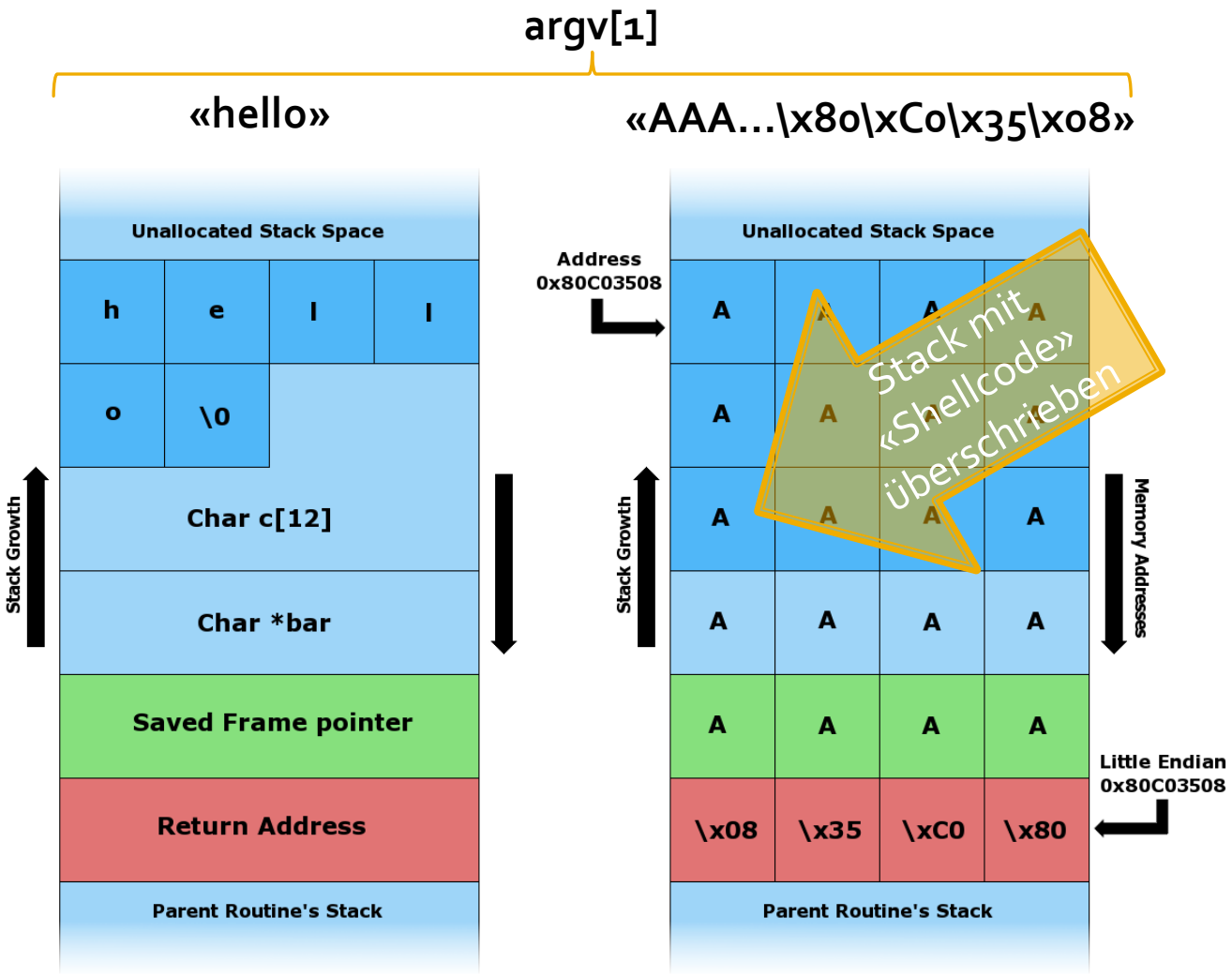
- Programmablauf als Kaskade von Funktionsaufrufen
- Orchestrierung via Call Stack und Calling Conventions



Nutzung einer Systemlücke

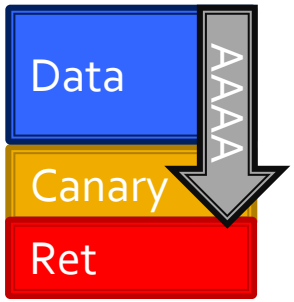
Code

```
#include <string.h>
void foo (char *bar)
{
  char c[12];
  strcpy(c, bar);
}
int main (
  int argc,
  char **argv)
  foo(argv[1]);
```



Abwehrmassnahmen

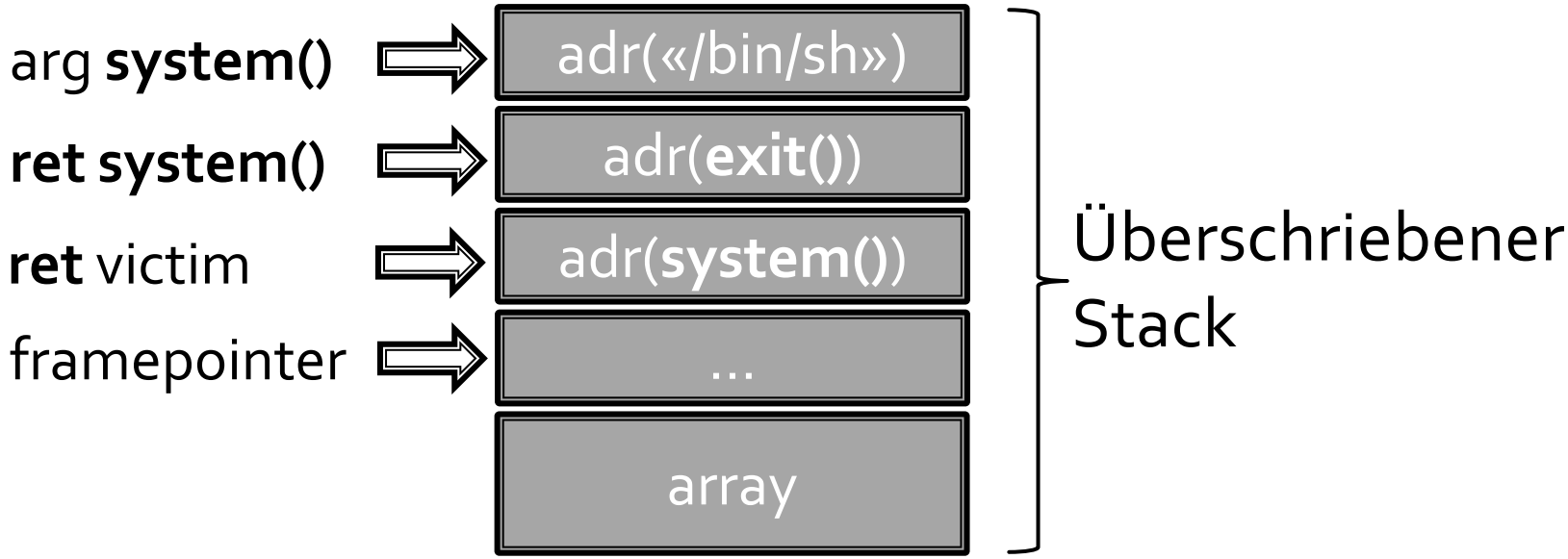
- «Canary» Flag zur Erkennung von überschriebenem Stack



- Alternatives Funktions-Aufrufsprotokoll
- Aufwärts- und abwärtswachsender Stack parallel unterhalten (Michael Franz, UCI)
- Execute ⊕ Write (genau eines von beiden) pro Speicherblock basierend auf dem No-Execute (NX) Modus des Prozessors

NX: Library Calls statt Shellcode

- Erwerb von *system* Privilegien via Wrapper
- Aufruf von **system()** zur Ausführung eines beliebigen Programmes



Abwehrmassnahmen

- Entfernung einzelner *libc* Funktionen
- Randomisierung der Startadressen der *libc* Funktionen
- Code signing

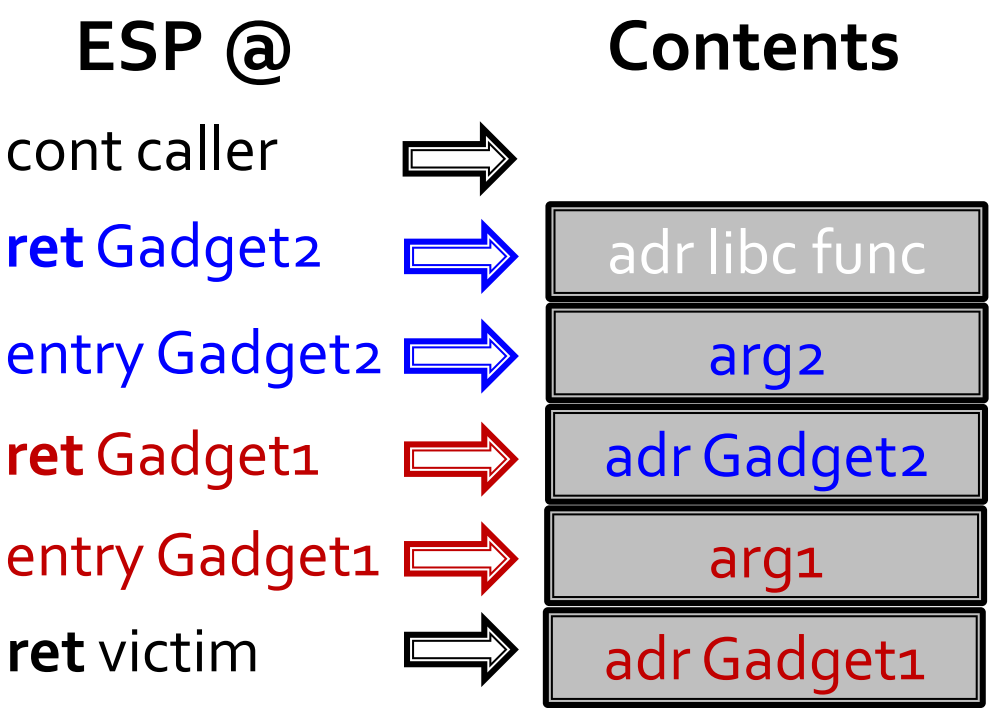
Return Oriented Programming

- (Missbräuchliche) Wiederverwendung von vorhandenem Library-Code (libc)
- Ganze Funktionen (z. B. **system()** Call) oder «Gadgets» der Form **instr1; instr2; …; return;**
  - Analog «head» in «the address»
  - f7 c7 07 00 00 00 of 95 45 c3 — test $0x00000007, %edi; setnzb -61(%ebp)
  - c7 07 00 00 00 of 95 45 c3 — movl $0x0f000000, (%edi); xchg %ebp, %eax; inc %ebp; ret

libc Call mit Register Parametern

- Gadget 1
  - **pop %ecx**
    ret
- Gadget 2
  - **pop %edx**
    ret

- Stack



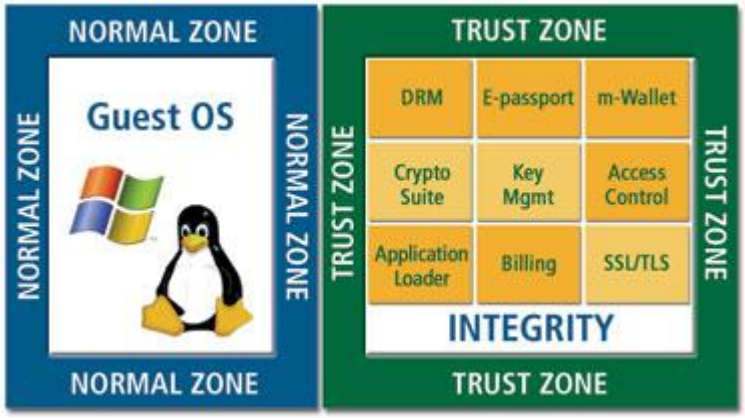
Abwehrmechanismen

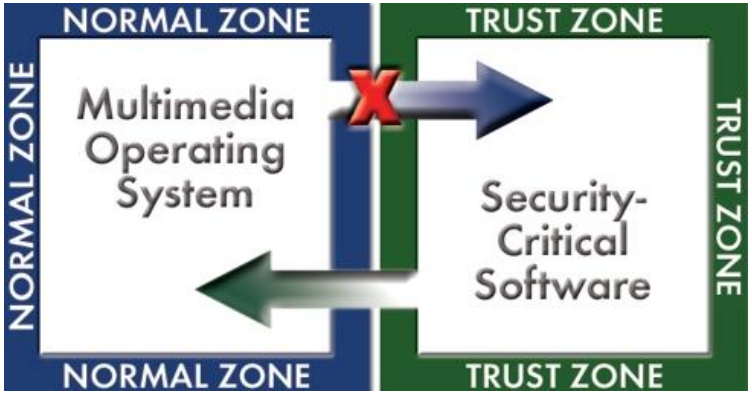
- Typsichere Sprachen
- Randomisierung
- Virtualisierung/ Managed Execution
- Sandboxes/ Geschlossene Systeme
- Custom Design (Hardware & OS)
- Duplexsystem wie in Flugzeugen
- Verfeinerte Privilegien («least privilege»)
  - «Trusted Execution Environments»

Sicherheit & Schutz 

Universalität

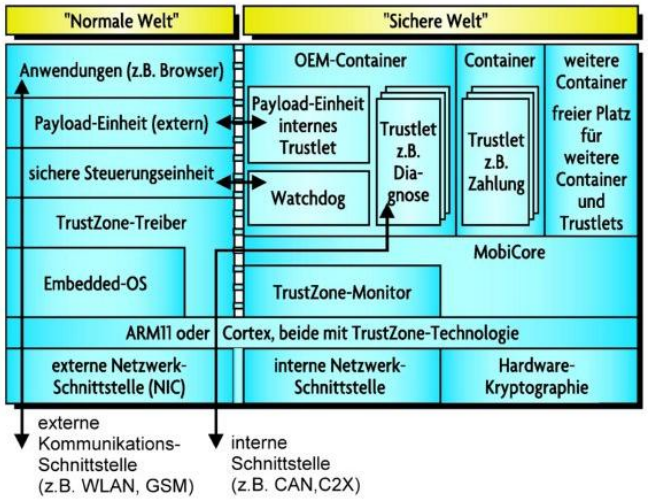
TRUST ZONE by ARM Konzept

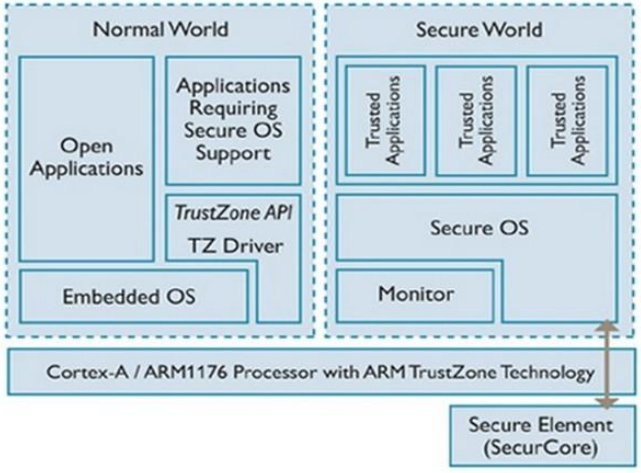




user /system *monitor* Modus

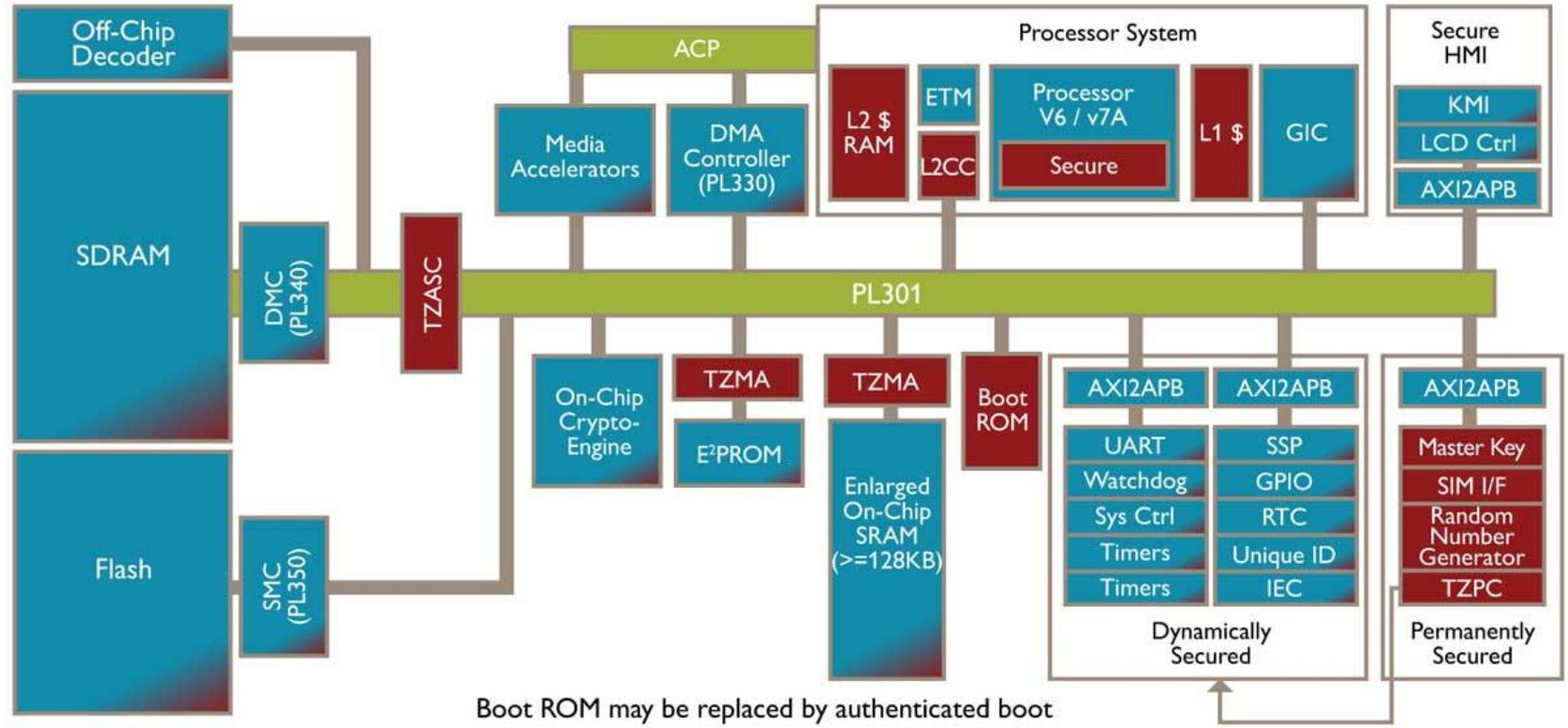
TEE





Software Architecture of TrustZone

TRUST ZONE Architektur



sicher/ trusted

normal/ untrusted